Human Motion Deblurring using Localized Body Prior

Jonathan Samuel Lumentut, Joshua Santoso, and In Kyu Park

Supplementary material

1 Foreground Discriminator for Pose Synthesizer

Our network structure for generating synthetic human motion blur dataset is motivated by Balakrishnan et al. [27]. We follow the network structure for both generator and discriminator. However, by utilizing only 1 discriminator as written in the original work [27] will forfeit some details on the human part. To handle this, we supplement 1 discriminator that focuses on the foreground part, and the details are shown in Table 1. The discriminator takes an input of a masked image, as shown in Figure 5 (a) in the manuscript. The mask is generated from the location of the detected body joint, and it is weighted with Gaussian distribution. Our generated dataset example is shown in Figure 1.

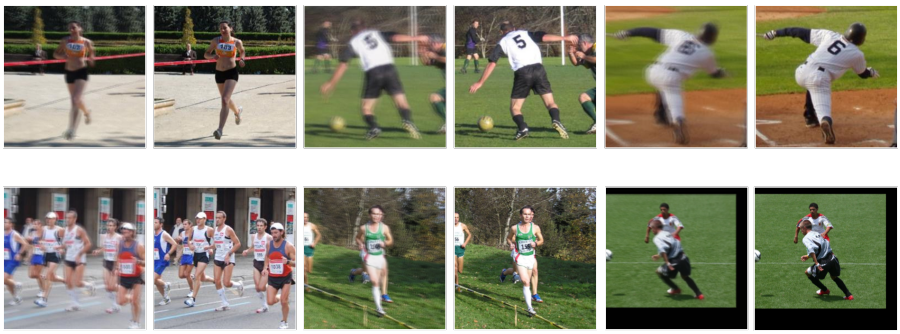

Fig. 1. Blurry human images and the corresponding ground truth in our dataset.

2 Deblurring Network

Our deblurring network is inspired by some state-of-the-art deblurring works [3, 13], where 9 ResNet Blocks (ResBlock) [23] are located in the middle of the structure to extract the features between convolution and deconvolution layers. The structure of our ResBlock is shown in Figure 2. The differences are: our network is employed with Instance Normalization (IN), which is excluded in [3] and no Dropout layers, compared to [13], that includes them. The detailed settings of our deblurring network are shown in Table 2.

Table 1. Detail of our additional foreground discriminator architecture. For pose synthesizer generator detail, please refer to the publication of Balakrishnan et al. [27].

Layer	Detail	Output size
Input (I_k^S)	-	$(H \times W \times 3)$
Conv 5×5	Leaky ReLU	$(H \times W \times 64)$
Max pooling	Max pooling 2D	$(H/2 \times W/2 \times 64)$
Concatenate 1	Concat	$(H/2 \times W/2 \times 92)$
Conv 5×5	Leaky ReLU	$(H/2 \times W/2 \times 128)$
Max pooling	Max pooling 2D	$(H/4 \times W/4 \times 128)$
Conv 3×3	Leaky ReLU	$(H/4 \times W/4 \times 256)$
Max pool 3	Max pooling 2D	$(H/8 \times W/8 \times 256)$
Conv 3×3	Leaky ReLU	$(H/8 \times W/8 \times 256)$
Max pooling	Max pooling 2D	$(H/16 \times W/16 \times 256)$
Conv 3×3	Leaky ReLU	$(H/16 \times W/16 \times 256)$
Max pooling	Max pooling 2D	$(H/32 \times W/32 \times 256)$
FC 1	ReLU	(256)
FC 2	Sigmoid	(1)

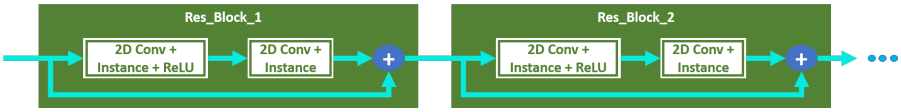

Fig. 2. The structure of our ResBlock that is concatenated until 9 times (9 ResBlocks).

3 3D Body Reconstruction Network

The 3D body reconstruction module is only used for extracting the body prior for human and scene regions localization. Thus, this network is frozen and we use the sophisticated model of the original authors [20]. For the detail of the network architecture, please refer to their publication [20].

4 Blurry Human Dataset

In generating the dataset, the multiple human poses are changed by varying the parameter, δ. The δ values are obtained from other videos relevant to the images in LSP [28] dataset. For example, to synthesize new poses of a human image (playing soccer in LSP [28]), we collect the poses of people playing soccer from YouTube, and the pose difference between each frame is collected as δ. By this approach, we utilize [27] to transform the current image with the current pose, θ_0, using the changed new pose, $\theta_0 + \delta$ (Fig. 5). In our implementation, this blurry image is added with translations in the image domain to provide additional camera motion.

Table 2. Detailed settings of our deblurring network architecture. The spatial output size is downscaled or upscaled according to the stride number.

Layer	Detail	Output size
Input (I^B)	-	$(H \times W \times 3)$
Conv 7×7	IN+ReLU	$(H \times W \times 64)$
Conv 3×3	IN+ReLU	$(H/2 \times W/2 \times 64)$
Conv 3×3	IN+ReLU	$(H/4 \times W/4 \times 128)$
Res_Blocks_1-9 3×3	IN+ReLU	$(H/4 \times W/4 \times 256)$
ConvTrans 3×3	IN+ReLU	$(H/2 \times W/2 \times 128)$
ConvTrans 3×3	IN+ReLU	$(H \times W \times 64)$
Conv 7×7	Tanh	$(H \times W \times 3)$

Table 3. Architecture details of Global (or Scene or Body) discriminator in our adversarial framework. Note that each task has each own discriminator.

Layer	Detail	Output size
Input (I^D or I^S)	-	$(H \times W \times 3)$
Conv 4×4	IN+LeakyReLU	$(H/2 \times W/2 \times 64)$
Conv 4×4	IN+LeakyReLU	$(H/4 \times W/4 \times 128)$
Conv 4×4	IN+LeakyReLU	$(H/8 \times W/8 \times 256)$
Conv 4×4	IN+LeakyReLU	$(H/16 \times W/16 \times 512)$
Conv 4×4	IN+LeakyReLU	$(H/16 \times W/16 \times 512)$
Conv 4×4	IN+Sigmoid	$(H/16 \times W/16 \times 1)$

5 Discriminator Networks

5.1 Global, Body, and Scene Discriminators

These discriminators have the same architectures because they process the same input, I^D and I^S. They are separated differently for solving each tasks as discussed in Section 3.2. Their detailed settings are represented in Table 3.

5.2 Patch Discriminator

The setting of this discriminator is only slightly different with others. 1 layer is reduced since this network only receives a cropped patch of 80×80 as discussed in Section 5. The detailed settings are shown in Table 4.

6 Additional Results

Additional deblurring results are animated in the attached video demo. Challenging cases are emphasized using red rectangular boxes. In visual comparison, we believe the closest work to us is by DeblurGAN-V2 [16]. However, their results are downgraded by the noise spots that appear in the deblurring result.

Table 4. Architecture details of Patch discriminator in our adversarial framework.

Layer	Detail	Output size
Input (P^D or P^S)	-	$(H \times W \times 3)$
Conv 4×4	IN+LeakyReLU	$(H/2 \times W/2 \times 64)$
Conv 4×4	IN+LeakyReLU	$(H/4 \times W/4 \times 128)$
Conv 4×4	IN+LeakyReLU	$(H/8 \times W/8 \times 256)$
Conv 4×4	IN+LeakyReLU	$(H/8 \times W/8 \times 256)$
Conv 4×4	IN+Sigmoid	$(H/8 \times W/8 \times 1)$

Our approach provides cleaner and better visual results than others, as shown in the emphasized region in the demo. Furthermore, we also provide finding that without the blurry human dataset, ours achieve PSNR/SSIM: 31.84/0.737 in GoPro test cases. Using it, we achieve higher results PSNR/SSIM: 32.51/0.805, as shown in the manuscript using *Ours-R*.